International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 15, No.1 (Apr-24)

http://dx.doi.org/10.12785/ijcds/1501110

Revolutionizing Cloud-Based Task Scheduling: A Novel Hybrid Algorithm for Optimal Resource Allocation and Efficiency in Contemporary Networked Systems

Punit Mittal[1], Dr. Satender Kumar[2] and Dr. Swati Sharma[3]

[1,2]*Department of Computer Science and Engineering, Quantum University, Roorkee, India*
[3]*Department of Information Technology, Meerut Institute of Engineering and Technology, Meerut, India*

Received 27 Jan. 2024, Revised 14 Mar. 2024, Accepted 18 Mar. 2024, Published 1 Apr. 2024

Abstract: The need for cloud computing has increased in the age of contemporary networked systems, driving the pursuit of optimal resource allocation and data processing. It is imperative in essential fields where security, such as transportation systems, depends on computing performance. Even after much research has been done on managing resources in cloud computing, finding algorithms that maximize job completion, minimize costs, and maximize resource consumption has remained a top priority. However, existing techniques have shown limitations, which calls for new ways. Our work shows the novel hybrid approach that has the potential to change the game completely. The Neural Network Task Classification (N2TC) is the result of merging neural networks with genetic algorithms. This ground-breaking method skillfully applies the Genetic Algorithm Task Assignment (GATA) for resource allocation while utilizing neural networks for task categorization. Notably, our algorithm carefully considers execution time, response time, costs, and system efficiency to promote fairness, a defense against resource scarcity. Our method achieves a remarkable 13.3% cost reduction, a stunning 12.1% increase in response time, and a 3.2% increase in execution time. These strong indicators act as a wake-up call, announcing our hybrid algorithm's power and revolutionary potential in transforming the paradigms around cloud-based task scheduling. This work represents a turning point in cloud computing, demonstrating an innovative combination of algorithms that not only overcomes current constraints but also ushers in a new era of efficacy and efficiency with far-reaching implications outside the domain of transportation systems.

Keywords: Cloud computing, Task Scheduling, Resource Allocation, Neural Network and Genetic Algorithm

1. INTRODUCTION

Recently, Cloud computing is a novel paradigm that maximizes efficiency while reducing management interactions [1]. This paradigm allows for distributed and parallel computing and is the foundation for processing large amounts of data, surpassing the capabilities of individual machines limited by RAM capacities [2][3]. Nevertheless, the exponential growth in cloud computing users and the increasing demands of contemporary technology have made resource allocation optimization in these environments even more crucial [4]. At the same time, the convergence of expanding vehicular networks and extensive cloud applications presents formidable cyber security challenges, given the limited space and computing capacity of these networks. The need for cloud services is growing daily, making resource management increasingly important. A scheduling algorithm based on a basic heuristic will not be employed to meet consumer demands. While greedy and genetic algorithms are effective in scheduling activities for neural network applications, there is one significant drawback to current methods: genetic algorithms have long execution times [5][6]. Our research proposes a comprehensive strategy to tackle this issue, which involves combining evolutionary algorithms and neural networks to reduce computing costs and optimize resource allocation [7][8].

Our suggested approach improves system efficiency and maximizes resource use in various computing paradigms. Our strategy guarantees comprehensive resource allocation and scheduling by placing emphasis on justice, optimal energy usage, minimized make span, load balancing, cost-effectiveness, and system efficiency [9][10]. Unlike previous methods, our approach uses neural networks to choose jobs for scheduling using genetic algorithms. It dynamically modifies resource allocation parameters for optimal utilization, adapting flexibly to changing cloud computing environments [11]. Additional research is required to ensure

E-mail address: punit.mittal06@gmail.com, deanacademics@quantumuniversity.edu.in swati.sharma@miet.ac.in

that jobs are allocated to cloud resources in an efficient manner, ultimately improving the quality of service criteria and respecting service levels [12].
Efficient resource scheduling not only facilitates the completion of activities in a minimal amount of time but also enhances the utilization ratio of resources, hence reducing resource consumption. The allotment of jobs has set off an important fact due to the rising burden felt by the cloud data center. This growth in workload has the potential to deplete cloud resources, therefore creating a shortage. Therefore, the area of cloud is now in its early stages, necessitating further investigation to effectively allocate jobs to cloud resources and achieve the goal of scheduling in order to enhance the quality of service criteria. The objective of scheduling is to determine the optimal allocation of resources for task execution. By doing so, scheduling procedures can improve various QoS parameters like task rejection ratio, resource utilization, reliability, energy consumption, and execution cost. It is important to achieve these improvements while ensuring that the service level agreement (SLA) is not compromised. Additionally, scheduling must take into account constraints such as deadlines and priorities and address the issue of load imbalance, which refers to situations where resources are either over utilized or underutilized.
The objective of our study are as follow:

- Meet the customer satisfaction in terms of execution time, response time, utilization, and cost.
- Improve the scalability to adapt the changes occur dynamically for the resources.
- Makes the use of neural networks to choose tasks for evolutionary algorithm scheduling, dynamically modifying resource allocation parameters for the best possible use in changing cloud computing environments.
- Integrates evolutionary algorithms and neural networks to lower computing costs and improve resource allocation, the research seeks to solve the difficulties associated with resource allocation optimization in cloud computing settings.

The major contribution of this paper are as follows:

- N2TC: Neural Network is used to classify the task into different classes then they are fed to Genetic Algorithm for scheduling. Our method can be used dynamically in different cloud environments because of the adaptability.
- GATA: Genetic Algorithm is used for scheduling of task to virtual machines. This approach is dynamically adapt the changes in demand of resources and done allocation optimally.
- Our methods works on various factor such as execution time, response time, utilization, and cost for the allocation of resources in data center.
- N2TC and GATA is used to improve the scalability to meet the customer satisfaction by allocating the optimal number of resources.

Furthermore, our N2TC and GATA provide insightful data that enables cloud providers to strategically allocate resources, thereby enhancing overall performance and cost efficiency [13]. Our methodology outperforms advanced methods, exhibiting notable improvements: a 3.2% reduction in execution time, a 13.3% cost reduction, and an impressive 12.1% enhancement in response time. Effective resource scheduling not only expedites task completion but also reduces resource consumption.

The organization of rest of the paper are as follows; Section 2, defines related work and provides a comprehensive analysis of existing literature and methodologies relevant to cloud computing and task scheduling. In section 3, we introduced - N2TC and GATA, Elaborates task classification using Neural Network with core principles behind the modified Genetic Algorithm-based task-scheduling algorithm. In section 4, discussing the configuration of the CloudSim simulator and the integration of the hybrid algorithm. Finally, in section 5, we discussed conclusion and future works of the paper.

2. Background Work

This section provides a brief synopsis of relevant works divided into two categories: dynamic resource allocation and meta heuristic-based resource scheduling.

A. Resource scheduling using heuristics

In [12], a Particle Swarm Optimization (PSO) algorithm has been proposed, which assigns jobs to virtual machines linked to physical data center equipment to maximize efficiency and prioritizes task scheduling in cloud computing depending on work length. In a heterogeneous machine-based data center, [14] used a parallel hybrid evolutionary algorithm with an island model for job migration, concentrating on energy-aware scheduling to minimize the makespan parameter. To reduce execution time and concentrate the fitness function on elitism and generation creation based on a threshold level, [15] presented a genetic algorithm that uses a roulette wheel selection approach. An integrated neural network and genetic algorithm technique was given by [16], to minimize processor context switching when managing simultaneous requests in a distributed system, particularly in a federated cloud setting. The Growable Genetic Algorithm (GGA) was presented by [17], combined a Random Multi-Weight algorithm with a Heuristic-based Local Search Algorithm (HLSA) to provide a growth stage to the genetic algorithm, allowing individuals to evolve along several growth paths. To augment the QoS parameters, [18][19] created a Directed Acyclic Graph (DAG) scheduling model. This model uses resource provisioning

and heuristic techniques to efficiently assign tasks to points and arrange the running sequence of jobs.

B. Dynamic resource allocation

A load balancing technique aiming to minimize idle time and makespan while limiting task migrations based on VM load and leveraging task priority levels for QoS in the cloud, simulated across sixteen data centers using Cloudsim [20]. An energetic dynamic resource allocation approach that considers arrival time and task size, enhancing response time, makespan, task completion ratio, and resource utilization. Some limitations are apparent in the research [21]. These include the lack of fairness concerns, which may result in task starvation, and the use of penalty functions in specific techniques, which may increase computing overhead. Furthermore, some methods allocate resources without considering crucial parameter balancing, which leads to less-than-ideal scheduling. Our suggested approach seeks to address these drawbacks.

C. Use of Genetic Algorithm

A crucial component of scheduling is managing the distribution of resources for incoming activities, especially in dynamic contexts where several jobs could occur at the same time. [22] developed a queuing method to effectively manage simultaneous task arrivals in order to overcome this difficulty. With this method, jobs are queued up and wait to be recalculated and reordered by the scheduling system. The system works by processing jobs in the queue one after the other, allocating them to the best resources possible through the use of a Genetic Algorithm (GA). Its main aim is to minimize execution time and maximize resource consumption at the same time. It provides an algorithm intended to address task scheduling difficulties by changing genetic algorithms' (GA) approaches. It also thoroughly examines GA and its application to job scheduling inside the Cloud computing area. The Max-Min technique is used by the algorithm to generate the first population. This strategy is purposefully meant to produce more optimum outcomes, especially in terms of *makespan*. The purpose of this change is to improve the efficacy and efficiency of work scheduling procedures in cloud computing settings [23]. The process of task scheduling is illustrated in Fig 1, whereby each user submits the tasks associated with their application, and the Cloud provider utilizes several methods to schedule these jobs. Optimization factors like minimal makespan, resource usage, and minimum cost are taken into consideration throughout the scheduling process. Heuristic techniques like GA, PSO, and ACO are frequently used to solve this optimization problem. These heuristic algorithms optimize for different parameters, as mentioned, providing workable solutions to the job scheduling puzzle in cloud computing settings. The comparison of various existing approach has shown in Table 1.

3. PROPOSED APPROACH

This section comprehensively describes our technique, including the algorithms and architectural structure that form its foundation. A hybrid cloud architecture that combined the features of public and private cloud services was used to implement our approach.

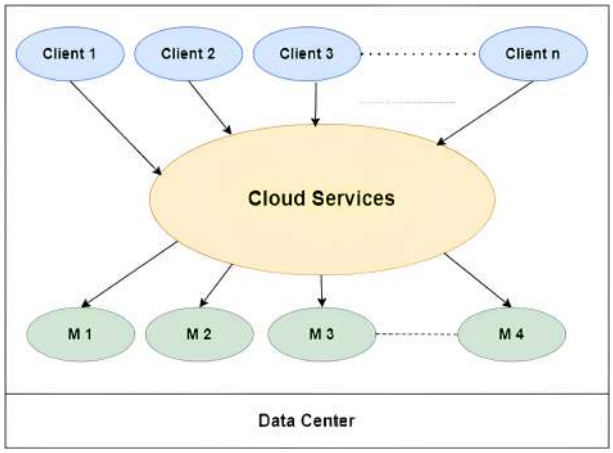

Figure 1. Task scheduling principles

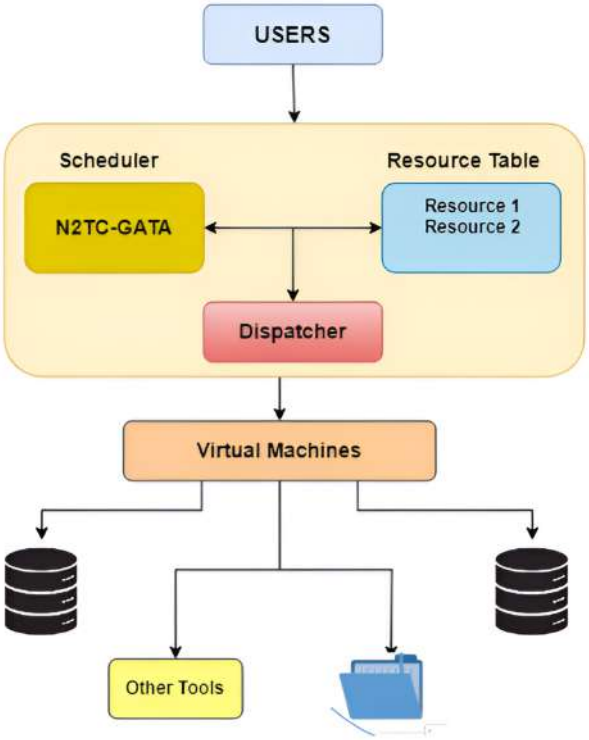

Figure 2. Overview of the architecture featuring

A. Framework

The best thing in cloud computing is effective work scheduling. Conventional schedulers are insufficient for this purpose due to the variety of user applications [24][25]. Fig. 2 shows the framework of our proposed method, which considers of 4 essential parts:

TABLE I. A Comparative study of existing method

Reference	Approach	Allocation	Strength	Drawback(s)
[12]	Multi Adaptive Learning for Particle Swarm Optimization	Static	Makespan, Load Balancing	Dynamic and and real-time scheduling are missing
[14]	Hybrid bi-objective discrete cuckoo search algorithm (HDCSA)	Dynamic	Makespan and Reliability	Other dominance state-of-the-art factors are missing
[15]	Hybrid Genetic Algorithm and Energy Conscious Scheduling	Dynamic	Minimized Makespan and Energy	Other dominance state-of-the-art factors are missing
[16]	RAA-PI-NSGAII	Static	Resource Utilization and Allocation	Time consuming process
[18]	Novel DAG scheduling model	Workflow	Makespan and Scheduling	Limited to 26 tasks
[19]	Hybrid heuristic	Workflow	Turnaround Time and Response Time	Only 28 task is used
[20]	Distributed load balancing algorithm	Workflow	Load Balancing Problem	Consider only small scale situations
[21]	Multiple Adaptive-resource-allocation Real-time Supervisor (MARS) scheme	Workflow	Response Time and Cost	The proposed method focus only IIoT

1) **Scheduler:** This module is the main engine for job scheduling, managing all computing operations in the system. It consists of two parts: N2TC and GATA.
2) **Dispatcher:** Dispatcher oversees the allocation of tasks to resources by evaluating resource conditions and allocating tasks appropriately.
3) **Resources:** This group includes all server-side programmers and tools that are used to process requests from clients, including file retrieval and computation.
4) **Resource Table:** Throughout the scheduling process, it is crucial to keep an eye on the availability of resources. This table displays the workload and recent status of each resource.

B. Model Mathematical

Procedures and formulas used in the N2TC and GATA module and two essential parts of the RAM module publicized in Fig. 2 are outlined in this section. In specific, we include formulas compulsory for task weighting, parameter-based categorization, and a fitness assessment. The acronyms that are shown in Table 2.

The task weight TW_i is determined by equation (1), which takes cost, execution time, and system efficiency characteristics into account. Depending on the state of the system, each parameter's starting weight is dynamically adjusted. Limited previous data is used to estimate parameter values for a new task entering the network [26][27].

$$TW_i = [WP(ET) * ET_i] + [WP(C)) * C_i] + [WP(SE) * SE_i] \quad (1)$$

Where $i = 1, 2, 3....n$ Whereas $WP(C)$ indicates the weight given to cost, $WP(SE)$ indicates the weight given to system efficiency, and $WP(ET)$ shows the weight given to execution time. Task i's execution time is shown by ET_i , its cost is indicated by C_i, and its system efficiency is indicated by SE_i.

$$|TW_i - CW_i| <\in\rightarrow TC_r = i \cup TC_r \quad (2)$$

In order to minimize the value of this function, equation (3) computes the fitness value for every task by taking response time and cost into account. Class size is denoted by $TC[r]$, class number is denoted by q, and task number is denoted by p. Task p within class q is indicated by TC[q,p]. F is a fairness-related parameter.

$$TC[q,p] = min[\sum_{q=1}^{r}\sum_{p=1}^{SC_r} RT(CT_{q,p} + C(TC_{q,p})) * F] \quad (3)$$

Equation (4) is used to increase the significance of fitness function while a task i from the prior scheduling is still in queue Q. The goal of this strategy is to increase the probability of carrying out tasks, which have been queued for the next iteration.

$$F_i = \begin{cases} 0.9 & \forall i \in Q \\ 1 & otherwise \end{cases} \quad (4)$$

Tasks with comparable weights can be grouped into distinct classes by using Equation (2), which compares a task's weight to the average weights of different task classes. The notation TW_i denotes the total weight of task i as determined by Equation (1), CW_r is the class r average weight, and TW_r is the total number of tasks in class r.

TABLE II. List of Abbreviation

Abbreviation	Meaning
CW	Class Weight
ET	Execution Time
F	Fairness
MR	Minimum Resources required
Q	Queue
r	Class Number (1,3)
RT	Response Time
SC	Size of Class
SE	System Efficiency
TC	Tasks of Class
TW	Weight of Task
WP	Weight of Parameter

C. Proposed Scheduling Algorithm

The method is a suggested scheduling procedure with an input task list. The method uses a three-class structure to classify tasks into distinct groups according to weighted features. If a job from the earlier scheduling step is still in the Q after this classification, its class rank is raised. For example, a job in the queue from the previous phase that belongs to class 2 is transferred to class 1. Following the assignment of each task to its proper class, the idle network resources and class 1 jobs are compared. If the quantity of idle resources is less than or equal to the total number of tasks in class 1, the class 1 tasks are directed to the GA.

Otherwise, there are also assignments from class 2. Tasks from class 3 are considered if idle resources remain after submitting assignments from classes 1 and 2. Because idle resources are finite, their availability is critical, and their quantity can change according to how many tasks are engaged at any particular point in time throughout execution. The chosen tasks are then fed into the GA. The fitness function value for every chromosome is calculated, and the first population is created at random. The gene's fitness function value for activities that were queued up from the last scheduling is increased by 10%. Next, chromosomes are arranged in descending order according to their fitness ratings.

The method selects parents iteratively by utilizing two-point crossover, modifying genes inside chromosomes, and applying elitism. Searches conducted locally around altered genes seek to find greater fitness values. This method continues until the iteration count hits the preset limit or an ideal task set is discovered. In terms of temporal complexity, the initial nested loop increases the total complexity by $O(n)$, where n is the number of jobs. The while loop's complexity, which is determined by the quantity of idle resources denoted by the symbol m, is $O(m)$. Sorting needs $O(nlogn)$ time, the initial population setup requires $O(n)$ time, and the final while loop has $O(kn)$ complexity. $O(nlogn+kn+m)$ is the total complexity of the time evaluation. The complexity if k and m are substantially smaller than n is $O(nlogn)$.

Algorithm: Optimized Task Scheduler

Input: Number of inputs t
Output: Optimal data

```
1. for each inputs in t
2.     for each class in [1,2,3]
3.         if(homogeneous inputs class)
4.             if(t in waiting and class > 1)
5.                 jump input to class (class - 1)
6.             else
7.                 write inputs in class
8. while(n free assets > inputs)
9.     inserting inputs from classes by scheduler
10. Initializing number of inputs
11. for each allele in the waiting queue
12.     Updates the allele with fitness
13. Sorting components as per fitness
14. while((!Optimal data) or (iteration!=max))
15.     Pick parent node highest fitness
16.     Pick two-point hybridization
17.     Transform allele within components
18.     Build adjacent search for betterment of allele
19.     Pick transformation
20. return optimal number of inputs
```

D. Neural Network Task Classification (N2TC)

1) **Neural network infrastructure:** N2TC is used for cloud computing input task classification. A feed-forward back-propagation neural network is utilized in our model.
2) **Data Preparation:** To begin, divide the data into three parts: seventy percent for training, 16% for network validation, and 16% more for network testing. While there isn't a set ratio, the traditional split of 70:30 is frequently accepted as standard [28]. Data for train tests and validation is chosen at random to maximize performance across these stages.

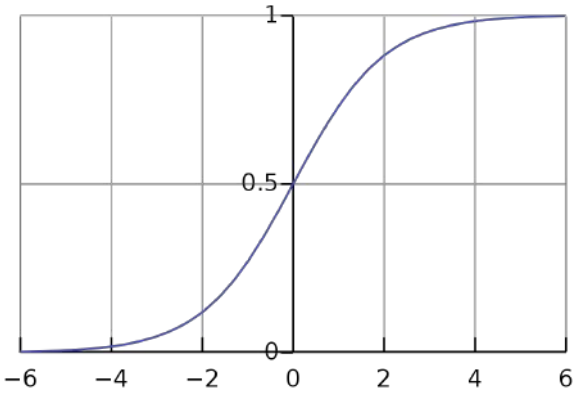

Figure 3. Equation-5 for transformation

3) **Transformation:** A derivative function, the sigmoid logarithmic transfer function, is frequently employed in back-propagation-trained multi-layer networks. Equation (5) is a typical illustration of this function:

$$A = \frac{1}{1 + e^{-net}} \tag{5}$$

Where:

- A represents the output of the sigmoid logarithmic transfer function.
- net is the result of the weighted sum of inputs: $\sum_1^n W_i * X_i$.
- X_i Denotes the input value.
- W_i signifies the weight associated with the respective input.

The sigmoid function, shown in Fig. 3, which translates input to a range between 0 and 1, is the activation function used in neural networks and is described by this equation. It helps with non-linear transformations and the learning of intricate patterns in the data.

4) **Training:** During training, we iteratively update the weights and biases using the scaled conjugate gradient approach. The following circumstances mark the end of the training process:
   - There are now more epochs than possible.
   - Time goes beyond the set upper bound.
   - Performance on the network declines below a given level.
   - The performance graph's gradient is below the minimum threshold.
   - Validation performance confirmation indicates a decrease from the last check.

5) **Function of Performance:** The efficiency function is used in Equation (6) to calculate the average squared errors between the target and the output. Declining network performance is the reason for the early termination of network training. Network training ends when there is a greater difference between the output and the aim.

$$performance = \frac{1}{n}\sum_{i=1}^{n}(Y^*(i) - Y(i))^2 \tag{6}$$

Memory reduction is another important factor taken into consideration in this network, with the goal of accelerating network execution. Although more layers result in improved network accuracy, they also increase run-time. Our thorough analysis revealed that the neural network's 20 hidden layers produced the best outcomes. High-priority jobs inside N2TC are sent to GATA and are divided into three segments. The N2TC task classification standards encompass execution time, cost, and system efficiency. Tasks from previous periods that are still in the waiting queue are advanced one level so they can be completed in order to prevent task hunger. Furthermore, in cloud computing, if there are more tasks designated as *first priority* than resources accessible, the lower priority activities are routed to GATA to optimize resource usage inside the network.

E. Genetic Algorithm Task Assignment (GATA)

Our method uses cloud computing resources to customize Genetic Algorithm Task Assignment (GATA) for task selection. We go from a binary to a decimal representation of chromosomes in order to maximize storage efficiency. Each gene on a chromosome stores a decimal number that represents the resources that are accessible. This paper proposes a modified Genetic Algorithm (TS-GA) to handle the problem of job scheduling in cloud computing settings. Its goals are to maximize resource consumption, minimize resource use costs, and improve job completion times on virtual machines (VMs). This suggested algorithm's selection approach inside the population is where the main novelty resides. In contrast to conventional techniques, our strategy keeps solutions that might not be selected for crossover but have excellent fitness. These fixes remain in the population and are added back in during later rounds. This approach has the benefit of preventing the premature discarding of potentially optimum solutions, which raises the possibility that the best solution will be found after several rounds.

Initial Population: The population initialization in the suggested TS-GA approach is carried out via random generation utilizing encoded binary (0, 1). This approach includes task-scheduling solutions represented as genes or chromosomes, where each gene or chromosome has the Virtual Machine (VM) ID and the associated ID for every job that is scheduled to run on these VMs. Each virtual machine (VM) and the tasks assigned to it are converted into binary bits as part of the encoding process. This encoding process serves as the foundation for the TS-GA technique, which is essential for improving task scheduling methods in cloud computing settings by enabling the systematic representation of VM-task assignments inside the chromosomal structure.

Selection Function: One notable feature of tournament

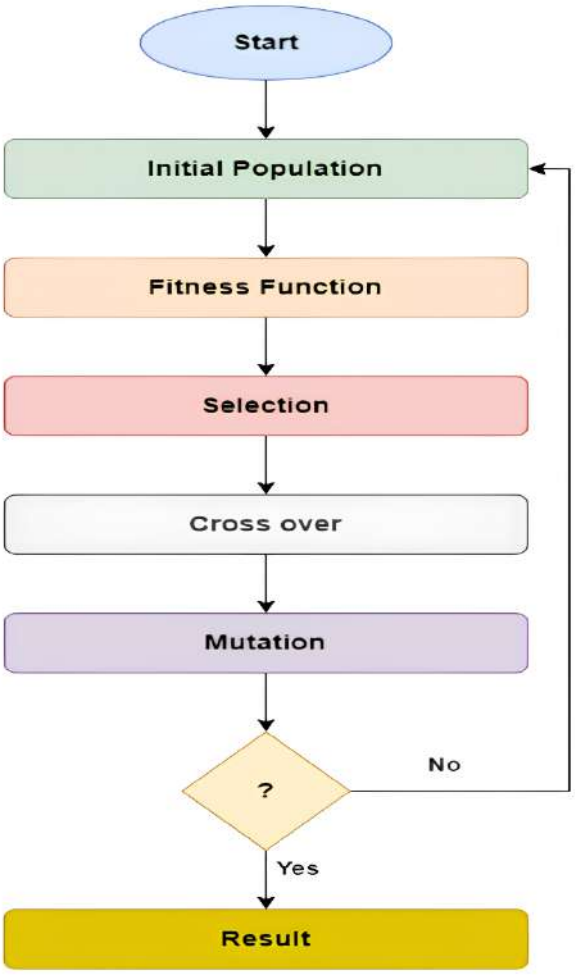

Figure 4. Genetic Algorithm

selection is its parallel implementation, appropriateness, and computational efficiency [29][30]. Tournament Selection is used in the developed TS-GA method to overcome the restriction related to population size limitations. Two people are chosen at random from the population in this approach. Next, a random integer between 0 and 1 is selected, represented by the letter r. The more fit of the two people is picked as a parent if $r < k$, where k is a preset threshold; if not, the less fit person is picked. The people who were not selected are subsequently reincorporated into the initial population and are still open to selection in later rounds. This technique reduces the restrictions imposed by population size limits while optimizing the selection process inside the genetic algorithm framework, improving its capacity to traverse complicated solution spaces efficiently.

Crossover: A unique crossover strategy used in the proposed TS-GA algorithm differs from the traditional crossover method used in the original GA. Here, the kids that come from the crossing process of two chromosomes are also considered as possible parents. Thus, this specific crossover mechanism produces four offspring from the crossover process. The two best children are then chosen from these four kids to be processed further by the algorithm. This novel crossover strategy increases the genetic variety and the likelihood of finding better solutions within the population, improving the algorithm's overall performance in navigating complex solution spaces.

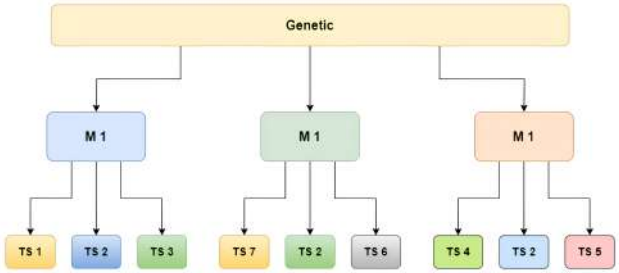

Figure 5. Representation tasks and VM

Mutation: In our scenario, the likelihood of a mutation is 5%. This change adds genetic material to the gene pool, which might lead to new gene values. These newly added gene values provide the genetic algorithm more exploration power, which increases the possibility of producing better results than previous iterations [31]. Mutation introduces genetic variation and provides a technique to escape local optima, which allows the algorithm to explore a more extensive solution space and maybe converge towards better answers.

Fitness Function: Task scheduling in the cloud is principally concerned with reducing the time needed to complete all tasks on the available resources as quickly as possible. Equation (7) [32] defines this completion time, $T_{i,j}$ for job i on VM_j.

$$T_{i,j} = S_j * C_i \tag{7}$$

Where $T_{i,j}$ taking into account i tasks and j virtual machines, denotes the maximum time for task i on VM_j to be completed. C_i is the Task's computational complexity, and S_j is the virtual machine's processing speed. It is necessary to schedule the execution times of all tasks among several virtual machines in order to minimize this completion time, which is represented by $T_{i,j}$. Equation (8) yields the processing time for a given job Pi on a chosen virtual machine VM_j:

$$ET_{i,j} = S_j * C_i C_i \tag{8}$$

In this case, Task P_i's processing time on VM_j is represented as $ET_{i,j}$. This calculation makes it possible to evaluate execution duration for various activities and virtual machines in detail, which serves as the foundation for efficient scheduling techniques meant to reduce completion times in cloud computing environments.

Equation (9) may be used to calculate the processing time for each job in a virtual machine [32].

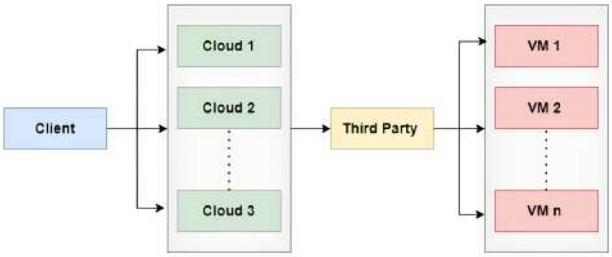

Figure 6. CloudSim Behavior [17]

$$fitness = \begin{cases} \sum_{i=1}^{n} RT_i + MR_i & Task_i \notin Q \\ 0.9 * \sum_{i=1}^{n} RT_i + MR_i & otherwise \end{cases} \quad (9)$$

The minor level of the fitness value inside the fitness function characterizes the chromosome's optimality. The necessary resources for the task i are indicated by MR_i. Tasks with lower resource requirements are more economical. Furthermore, RT_i represents the response time for task i, which should be as small as possible. Once each gene's fitness function has been established, the fairness parameter determines if task i has remained in the queue from an earlier schedule. The fitness value of tasks still in the queue is increased by 10%, giving them more opportunities to get resources and reducing the likelihood of famine. The goal of this intervention is to keep resource allocation equitable.
Optimal Solution: Some solutions in the suggested TS-GA method may have the best fitness functions but have yet to be chosen in the crossover phase. A novel aspect of this strategy ensures that these answers are not removed from the population; instead, they are specifically selected and reintroduced once the subsequent iteration begins. This tactical move is essential because it keeps potentially valuable solutions in the population, raising the possibility that the optimal solution may be found in later rounds. The algorithm maintains variety and the possibility of finding the best optimum solution across several iterations by keeping and reincorporating these intriguing options.

The University of Melbourne has published the CloudSim toolbox, which helps researchers simulate cloud computing systems. CloudSim offers essential elements for modelling and simulating many aspects of cloud computing infrastructures. Within CloudSim, users send requests in the form of cloudlets, distinguished by attributes like the size of the file and the number of instructions to be performed. After that, the broker receives these cloudlets and schedules them into Virtual Machines (VMs) per the predetermined scheduling policies. Because CloudSim uses broker-driven regulations, scheduling tactics may be customized and flexible. Virtual machines that may be constructed on hosts are represented by the VM class in CloudSim. Most importantly, CloudSim has a data center component that can support a maximum number of hosts. The broker dynamically sets up the hosts and virtual machines (VMs), enabling flexibility in response to changing simulation needs [33]. For academics in the area, CloudSim's architecture is a vital resource since it offers a thorough and adaptable framework for simulating and assessing cloud computing systems.
Tools used:

- **CloudSim:** To describe and simulate cloud settings, we used CloudSim, an open-source simulation tool for cloud computing. We were able to establish virtualized data centers, hosts, virtual machines, and scheduling policies with the help of CloudSim. To correctly mimic the behavior of real-world cloud systems, we modified CloudSim.
- **TensorFlow and Keras:** We used the TensorFlow and Keras libraries to create the neural network-based task classification (N2TC) module. These tools provided us with an efficient processing capability to handle massive datasets, as well as a high-level interface for creating and training deep learning models.
- **Genetic Algorithm Implementation:** For job assignment, we created a custom genetic algorithm (GATA) implementation. This approach offered flexibility in creating crossover, mutation, and selection procedures customized to our study goals, and it made use of Python's NumPy module for numerical computations.

Configuration: To replicate a multi-tenant cloud system with diverse virtual machines and workload patterns, we set up CloudSim. We provide specifications for things like network properties, job scheduling methods, and VM provisioning procedures.
Workload Generation: We created a workload generator module inside of CloudSim to provide artificial workload traces. This module used statistical distributions drawn from real-world data to produce resource needs and task arrivals.

4. EVALUATION AND EXPERIMENT RESULTS

To assess the performance of our suggested method in a range of workload scenarios and system configurations, we ran a number of tests. We experimented with many variables to see how they affected system performance, including workload intensity, virtual machine capacity, and scheduling strategies.
We use the Google cluster-traces v3 dataset [29] for evaluation purposes. This dataset needs more detailed information about end users, their data, or storage systems and instead focuses primarily on resource demands and utilization. There are 405,894 rows of data in it. The dataset includes workload statistics from a variety of applications, such as databases, web servers, and scientific computing workloads, spanning many months. Millions of task submissions with different priority, resource needs, and execution durations are contained in it. To eliminate outliers, normalize features, and deal with missing values, we preprocessed the datasets before utilizing them. To aid in the creation and assessment of the model, we further divided the data into training, validation, and testing sets. Table 3 outlines the hardware

setup used to carry out the suggested method. Our method is predicated on several suppositions, including:

- **Task Independence:** Each task in the system runs separately from the others. Task i can be carried out independently of task j's prior completion.
- **Lack of Deadlines:** There are no set deadlines for tasks to be completed.
- **Non-Preemption of Tasks:** In our network, tasks are not preemptible, meaning that resources are allotted and reserved until the job is finished.

Performance measures: They used measures like makespan, fairness index, throughput, and resource consumption to gauge performance. These measurements gave us information about the effectiveness, scalability, and equity of our scheduling algorithms.

When a job was first added to the network, it was queued up for scheduling. We chose ten jobs to be executed based on predetermined criteria throughout the scheduling process. We evaluated our technique using this collection of 10 tasks, which we thought would be sufficient to gauge its effectiveness. Within the framework of our genetic algorithm, every gene on a chromosome is associated with a particular purpose. Research like [31][32] has frequently noted that genetic algorithms function best when only a few genes are on each chromosome. However, it is essential to recognize that chromosomal size increases the algorithm's computational complexity, placing restrictions on the system because of higher processing requirements.
Metrics relevant to a task, like cost, execution time, and reaction time, can vary. This section shows that the task set selected by our model is better than those selected by other methods. As indicated, N2TC obtains tasks and classifies them according to parameters, including execution time, system efficiency, and cost.

The graph of network performance is shown in Fig 7. After 32 epochs, the graph ends, with epoch 26 showing the greatest performance.

A test graph with a more significant curvature indicates a higher probability of overfitting in the network. A graph with a decreasing trend indicates excellent network performance. Given GATA's findings, the fitness function is essential to the genetic algorithm's architecture. Then, class 1 tasks and sometimes class 2 tasks are sent to GATA so that it can select the best possible set of tasks to be completed. The perfect set consists of 10 selected tasks. The fitness function's trajectory during determining this ideal task set is depicted in Fig. 7. Within a chromosome, minimized fitness values are correlated with task set optimality. The graph's downward trend indicates that the GATA arrangement is appropriate. We then compared our suggested method and well-known algorithms like Shortest Job First (SJF) and First in First out (FIFO). For comparative analysis, these

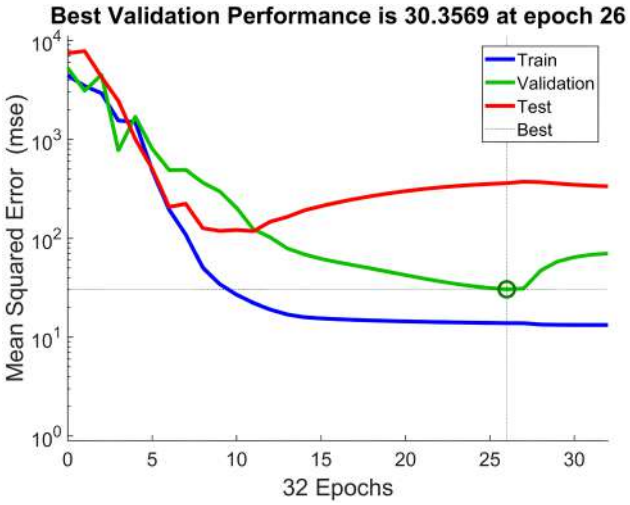

Figure 7. Network Performance during the Validation, Testing, and Training Stages Using cross-entropy to evaluate the model's development over a 32-epoch period in terms of individual epochs.

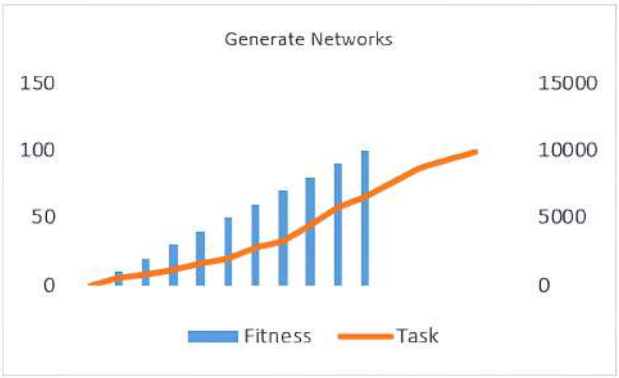

Figure 8. The Development of Task Selection Fitness Values throughout Generations

benchmark methodologies provide insightful information. The task execution times for each of the five techniques are shown in Fig 9. Our suggested approach shows generally decreased execution times across tasks for the 10 selected tasks for each strategy.

The system utilization rate, a critical indicator of how best to use the network's resources, is shown in Fig. 10. An optimal operating state is characterized by a greater utilization rate, which indicates maximum utilization and minimal idle resources. The approach suggested by [14] has the lowest utilization rate of all the approaches under consideration; in contrast, our method has the second-highest utilization rate, highlighting its effectiveness in resource allocation and consumption.
A comparison of the expenses related to task execution for each of the five techniques is shown in Fig. 11. Interestingly, when adding up the expenses of each action, our suggested method shows the lowest total cost for carrying out all 11 chores.

TABLE III. Specifications of the platform utilized

Abbreviation	Meaning
Computer	Dell
RAM	32 GB
OS	64-bit, Windows 10
CPU	Intel® Core™ i7-3230M CPU @ 2.60 GHz

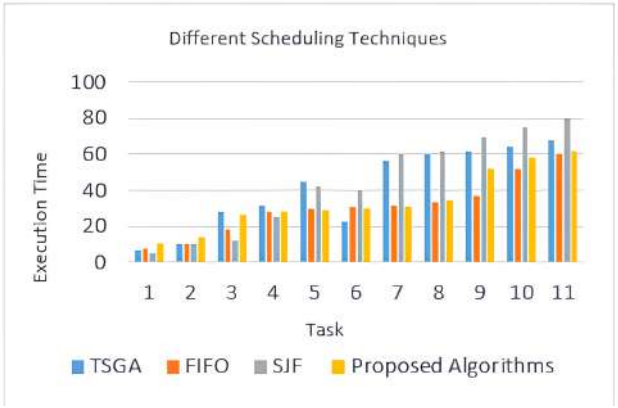

Figure 9. Ten Scheduled Tasks with Comparative Execution Times Using Different Methods

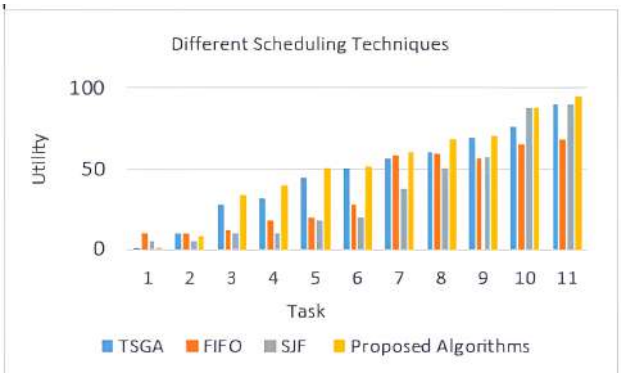

Figure 10. Comparative Rates of System Utilization for Different Scheduling Techniques

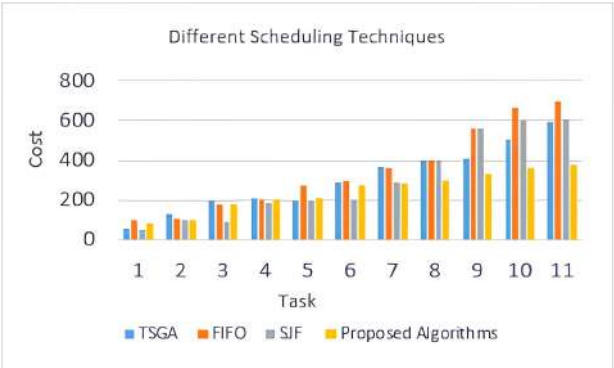

Figure 11. Comparative Costs of Using Different Methods to Complete Various Tasks

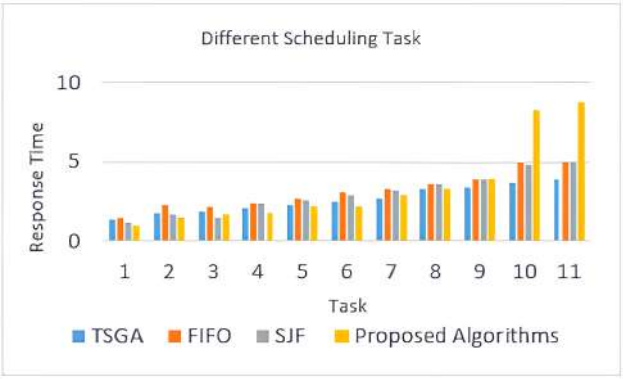

Figure 12. Comparison of Response Times for Various Tasks Using Different Scheduling Methods

The response time graph for each of the five scheduling methods is shown in Fig. 12, showing the time between when a job is submitted to cloud computing and when the user receives their first network response.

When compared, our approach shows the fastest response time for the majority of tests, except task 10. It is clear from a collective analysis of the supplied graphs that the suggested method performs better than the other options. Because of its exceptional performance in various measures, it is a good option for a wide range of applications. The performance evaluation of the previously described solutions is summarized in Table 4, which also shows the overall average of the ten chosen tasks for each solution. The values for each job have been transformed to a range between 0 and 1 to simplify the results. Notable improvements are obtained by the suggested method compared to the average performance of current advanced methods. In particular, improvements of about 3.2% in response time, 13.3% in expenses, and 12.1% in execution time have been noted.

Fig. 13 shows that nine out of 10 tasks, the suggested solution outperforms alternative approaches in terms of response time. It also shows the most economical implementation of all the techniques for each of the ten tasks. Although our solution ranks second in utilization rate, it is the best at optimizing execution time. The usefulness of the suggested model in reducing task famine, increasing task selection, and strengthening these previously mentioned characteristics is supported by graphic findings. As a result, the suggested strategy outperforms the previously listed techniques, demonstrating its overall superiority.

TABLE IV. Comparison of Performance Evaluation of Various Algorithms

Methods	Cost Response	Time	Utility	Execution Time
FIFO	0.311	0.28	0.233	0.36
HDCSA [14]	0.441	0.357	0.118	0.41
SJF	0.262	0.296	0.279	0.45
MOHGA [15]	0.302	0.508	0.546	0.31
PROPOSED APPROACH	0.279	0.233	0.475	0.12

Figure 13. Performance Summary of existing and proposed approach parameters

Evaluation Method: To confirm the resilience of our strategy, we carried out a thorough cross-validation and sensitivity analysis. To prove the superiority of our suggested approach, we measured its performance against baseline algorithms like first-come, first-served, SJF etc. scheduling.

Analysis: To evaluate the experimental outcomes, we used qualitative judgments, statistical testing, and visualization methods. We found patterns, trends, and trade-offs in the way the system performed under various circumstances, providing information about the advantages and disadvantages of our method.

5. CONCLUSION AND FUTURE SCOPE

In conclusion, resource allocation in the cloud is still a significant problem that requires constant research into workable solutions. Heuristic techniques are becoming increasingly common in more extensive contexts, which is evidence of their effectiveness. Our suggested method addresses resource assignment difficulties in cloud computing by combining neural networks and evolutionary algorithms in a novel way to achieve optimum solutions. The overall objective of the developed TS-GA is to reduce completion time and cost while optimizing resource use. Significantly, lower completion times highlight the algorithm's effectiveness. These results highlight how well the algorithm handles resource allocation problems, suggesting that it may be used in real-world scenarios and further developed for various uses. In comparison to HDCSA and other benchmark algorithms, we found that our approach produces a greater system utilization rate, Improved Performance, Cost Effectiveness, Task Selection Optimization and General Adaptability. This shows that our method efficiently reduces idle resources and maximizes resource use, improving system efficiency as a whole. The main limitations of the suggested work is that, this algorithm would not work well where workflows required highly inter-process communication. For this reason, the scenarios for highly communication-intensive workflows have been purposefully left out of this paper. Although extensive, the Google cluster-traces v3 dataset does not fully cover all facets of actual cloud computing infrastructures. Additional datasets that offer more in-depth details about end users, data properties, and storage technologies may be gathered and analyzed in future research. Future research should focus on addressing privacy and security issues associated with job scheduling in cloud systems. Creating scheduling algorithms that protect privacy and making sure sensitive data is secure while tasks are being completed are important factors to take into account before implementing cloud computing technology. Future versions of this approach could manage more tasks with dependencies, prioritize tasks with time limits, and improve load balancing within predetermined bounds. Moreover, the suggested cloud-based scheduling model expands its possible uses beyond cloud computing, becoming especially pertinent in strengthening cybersecurity in transportation networks. This strategy, which uses the speed and flexibility of cloud computing, can help with increased security monitoring and proactive responses to cyberattacks on transportation infrastructure.

REFERENCES

[1] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," Gaithersburg, MD, USA, Tech. Rep., 2011.

[2] P. Gao, Z. Han, and F. Wan, "Big data processing and application research," in *2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM)*, 2020, pp. 125–128.

[3] M. Manavi, Y. Zhang, and G. Chen, "Resource allocation in cloud computing using genetic algorithm and neural network," 2023.

[4] A. Belgacem, S. Mahmoudi, and M. Kihl, "Intelligent multi-agent reinforcement learning model for resources allocation in cloud computing," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, Part A, pp. 2391–2404, 2022.

[5] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng, and G. Min, "Computation offloading and task scheduling for dnn-based applications in cloud-edge computing," *IEEE Access*, vol. 8, pp. 115 537–115 547, 2020.

[6] D. Cui, Z. Peng, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for grid or iaas cloud," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1030–1039, 2020.

[7] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840–1852, 2021.

[8] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1145–1168, Jun 2021. [Online]. Available: https://doi.org/10.1007/s11036-020-01624-1

[9] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," *Optik*, vol. 258, p. 168677, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402622000936

[10] G. U. Srikanth and R. Geetha, "Effectiveness review of the machine learning algorithms for scheduling in cloud environment," *Archives of Computational Methods in Engineering*, vol. 30, no. 6, pp. 3769–3789, Jul 2023. [Online]. Available: https://doi.org/10.1007/s11831-023-09921-0

[11] H. Godhrawala and R. Sridaran, "Improving architectural reusability for resource allocation framework in futuristic cloud computing using decision tree based multi-objective automated approach," in *Advancements in Smart Computing and Information Security*, S. Rajagopal, P. Faruki, and K. Popat, Eds. Cham: Springer Nature Switzerland, 2022, pp. 397–415.

[12] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 4, pp. 4313–4327, Apr 2023. [Online]. Available: https://doi.org/10.1007/s12652-023-04541-9

[13] J. Xing, "Network security optimization method based on genetic algorithm," in *Innovative Computing*, J. C. Hung, J.-W. Chang, Y. Pei, and W.-C. Wu, Eds. Singapore: Springer Nature Singapore, 2022, pp. 1359–1366.

[14] Y. Asghari Alaie, M. Hosseini Shirvani, and A. M. Rahmani, "A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach," *The Journal of Supercomputing*, vol. 79, no. 2, pp. 1451–1503, Feb 2023. [Online]. Available: https://doi.org/10.1007/s11227-022-04703-0

[15] G. Agarwal, S. Gupta, R. Ahuja, and A. K. Rai, "Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog–cloud computing," *Knowledge-Based Systems*, vol. 272, p. 110563, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123003131

[16] J. Chen, T. Du, and G. Xiao, "A multi-objective optimization for resource allocation of emergent demands in cloud computing," *Journal of Cloud Computing*, vol. 10, no. 1, p. 20, Mar 2021. [Online]. Available: https://doi.org/10.1186/s13677-021-00237-7

[17] G. Zhou, W. Tian, R. Buyya, and K. Wu, "Growable genetic algorithm with heuristic-based local search for multi-dimensional resources scheduling of cloud computing," *Applied Soft Computing*, vol. 136, p. 110027, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494623000455

[18] R. Rajak, S. Kumar, S. Prakash, N. Rajak, and P. Dixit, "A novel technique to optimize quality of service for directed acyclic graph (dag) scheduling in cloud computing environment using heuristic approach," *The Journal of Supercomputing*, vol. 79, no. 2, pp. 1956–1979, Feb 2023. [Online]. Available: https://doi.org/10.1007/s11227-022-04729-4

[19] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *Journal of Cloud Computing*, vol. 7, no. 1, p. 4, Feb 2018. [Online]. Available: https://doi.org/10.1186/s13677-018-0105-8

[20] A. Semmoud, M. Hakem, B. Benmammar, and J.-C. Charr, "Load balancing in cloud computing environments based on adaptive starvation threshold," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 11, p. e5652, 2020, e5652 cpe.5652. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5652

[21] Y. Shin, W. Yang, S. Kim, and J.-M. Chung, "Multiple adaptive-resource-allocation real-time supervisor (mars) for elastic iiot hybrid cloud services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1462–1476, 2022.

[22] R. S and N. Ealai Rengasari, "Dynamic scheduling of data using genetic algorithm in cloud computing," *INTERNATIONAL JOURNAL OF COMPUTING ALGORITHM*, vol. 2, pp. 11–15, 06 2013.

[23] S. Singh and M. Kalra, "Scheduling of independent tasks in cloud computing using modified genetic algorithm," in *2014 International Conference on Computational Intelligence and Communication Networks*, 2014, pp. 565–569.

[24] A.-N. Zhang, S.-C. Chu, P.-C. Song, H. Wang, and J.-S. Pan, "Task scheduling in cloud computing environment using advanced phasmatodea population evolution algorithms," *Electronics*, vol. 11, no. 9, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/9/1451

[25] S. Shukla, A. K. Singh, and V. Kumar Sharma, "Survey on importance of load balancing for cloud computing," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021, pp. 1479–1484.

[26] L. Shi, J. Xu, L. Wang, J. Chen, Z. Jin, T. Ouyang, J. Xu, and Y. Fan, "Multijob associated task scheduling for cloud computing based on task duplication and insertion," *Wireless Communications and Mobile Computing*, vol. 2021, p. 6631752, Apr 2021. [Online]. Available: https://doi.org/10.1155/2021/6631752

[27] V. K. Sharma, S. Sharma, M. Rawat, and R. Prakash, *Adaptive Particle Swarm Optimization for Energy Minimization in Cloud: A Success History Based Approach*. Singapore: Springer Nature Singapore, 2023, pp. 115–130. [Online]. Available: https://doi.org/10.1007/978-981-99-6034-7_7

[28] B. Tutumlu and T. Saraç, "A mip model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting," *Computers Operations Research*, vol. 155, p. 106222, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054823000862

[29] J. Wilkes, "Google cluster-usage traces v3," Google Inc., Mountain View, CA, USA, Technical Report, Apr. 2020, posted at https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

[30] M. Gupta, P. Singh, and V. K. Sharma, "Loan eligibility prediction model using machine learning algorithms," in *Artificial Intelligence, Blockchain, Computing and Security Volume 2*. CRC Press, 2024, pp. 20–26.

[31] L. Jia, L. Yisheng, S. Ying, and L. Jian, "Solving resource-constrained project scheduling problem via genetic algorithm," *Journal of Computing in Civil Engineering*, vol. 34, no. 2, p. 04019055, Mar 2020. [Online]. Available: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000874

[32] P. Devarasetty and S. Reddy, "Genetic algorithm for quality of service based resource allocation in cloud computing," *Evolutionary Intelligence*, vol. 14, no. 2, pp. 381–387, Jun 2021. [Online]. Available: https://doi.org/10.1007/s12065-019-00233-6

[33] W. Khan, M. Kadri, and Q. Ali, "Optimization of microchannel heat sinks using genetic algorithm," *Heat Transfer Engineering - HEAT TRANSFER ENG*, vol. 34, 01 2013.

Punit Mittal is the Research Scholar in the Department of Computer Science and Engineering, Quantum University, Roorkee, India. Currently, He is pursuing a Ph.D. in Computer Science and Engineering. He has completed his M.Tech (IIITM Gwalior, India) and B.Tech in Computer Science and Engineering. He has published many research papers in reputed Scopus-Indexed journals, IEEE, and Springer Conference. His area of expertise is Machine Learning and Cloud Computing.

Dr. Satender Kumar is the Dean Academic of Quantum University and Head of the Department of Computer Science & Engineering and Computer Application Department. He has more than 19 years of experience in teaching. He has published 16 research articles in National and International peer reviewed Journals, authored 3 books contributed 3 book chapters and owned 9 patents.

Dr. Swati Sharma is the Professor and Head in the Department of Information Technology, Meerut Institute of Engineering and Technology, Meerut, India. She has completed Ph.D. in Computer Science. An academician with more than 13 years of teaching experience authored more than a dozen of research papers in reputed SCI, Scopus-indexed journals, International Journal, IEEE Conferences, patent and copyright to her credit. Her areas of interest are machine learning, data science, and deep learning. She is currently doing research in the area of Data Analysis.